Homework 3: Guessing Game
Adopted from CSE 142
Due: Wednesday, July 20, 2016

This assignment will give you practice with while loops and pseudorandom numbers. You are going to write a program that allows the user to play a simple guessing game in which your program thinks up an integer and allows the user to make guesses until the user gets it right. For each incorrect guess you will tell the user whether the right answer is higher or lower. Your program is required to *exactly* reproduce the format and behavior of the log of execution at the end of this write-up.

At a minimum, your program should have the following functions:

- A method that introduces the game to the user
- A method to play one game with the user (just one game, not multiple games)
- A method to report overall results to the user

You may define more functions than this if you find it helpful, although you will find that the limitation that methods can return only one value will tend to limit how much you can decompose this problem.

You are to define a global variable/class constant for the maximum number used in the guessing game. The sample log shows the user making guesses from 1 to 100, but the choice of 100 is arbitrary. By introducing a constant for 100, you should be able to change just the value of the constant to make the program play the game with a range of 1 to 50 or a range of 1 to 250 or some other range starting with 1.

When you ask the user whether or not to play again, you should use the raw_input() function. You should continue playing if this answer begins with the letter “y” or the letter “Y”. Notice that the user is allowed to type words like “yes”. You are to look just at the first letter of the user’s response and see whether it begins with a “y” or “n” (either capitalized or not) to determine whether to play again.

Assume that the user always types an integer when guessing, that the integer is always in an appropriate range and that the user gives you a one-word answer beginning with “y”, “Y”, “n” or “N” when asked whether to play again. You may assume that no game involves more than 9,999 guesses.

You will notice at the end of the log that you are to report various statistics about the series of games played by the user. You are to report the total number of games played, the total number of guesses made (all games included), the average number of guesses per game, and the best (fewest) number of guesses used in any single game. The average number of guesses per games should be rounded to one decimal place (you can use either the round() function or a printf).

Because this program uses pseudorandom numbers, you won’t be able to recreate the sample log. We will provide sample logs where the answer is always 42. Obviously you won’t want your program to always pick 42 as the number to be guessed. You should modify your program to set the answer to 42 and check its behavior against the sample logs. Then you should put it back to the normal behavior of picking a different number for every game before you turn it in.

Here are a few helpful hints to keep in mind.

- To deal with the yes/no response from the user, you will want to use some of the String functions. Please refer to your book.
- It’s a good idea to change the value of your class constant and run the program to make sure that everything works correctly with the new value of the constant. For example, turn it into a guessing game for numbers between 1 and 5.
- While you are developing your program, you might want to have it print out the answer before the user begins guessing. Obviously you don’t want this in the final version of the program, but it can be helpful for you while you are developing the code.
-

You should handle the case where the user guesses the correct number on the first try. Print the following message:

```
You got it right in 1 guess
```

This program is more difficult to decompose into methods, so you may end up having functions that are longer than 15 lines. You can also include more code in your main module than we allowed in the last program. In particular, you are required to have a while loop in main that plays multiple games and prompts the user for whether or not to play another game. You shouldn't have all of the code in main because you are required to have the functions described at the beginning of this write-up.

We will once again expect you to use good programming style and to include useful comments throughout your program. We will expect you to make appropriate choices about when to store values as int versus double, which if/else constructs to use, what parameters to pass, and so on.

You are limited to language features we have discussed so far in the class. You are not allowed to use break statements.

Use whitespace and indentation properly. Limit lines to 100 characters. Give meaningful names to methods and variables, and follow Python's naming standards. Localize variables whenever possible. Include a comment at the beginning of your program with basic description information and a comment at the start of each function. Some students try to achieve repetition without properly using while loops, by writing a method that calls itself; feel free to use recursion but remember that we have not discussed this in class.

Your program should be stored in a file called Guess.py.

Log of execution (user input bold and underlined)

```
This program allows you to play a guessing game.
I will think of a number between 1 and
100 and will allow you to guess until
you get it.  For each guess, I will tell you
whether the right answer is higher or lower
than your guess.

I'm thinking of a number between 1 and 100...
Your guess? 50
It's lower.
Your guess? 25
It's lower.
Your guess? 12
It's lower.
Your guess? 6
You got it right in 4 guesses
Do you want to play again? y

I'm thinking of a number between 1 and 100...
Your guess? 50
It's lower.
Your guess? 25
It's lower.
Your guess? 12
It's higher.
Your guess? 18
You got it right in 4 guesses
Do you want to play again? YES

I'm thinking of a number between 1 and 100...
Your guess? 50
It's higher.
Your guess? 75
It's lower.
Your guess? 62
It's higher.
Your guess? 68
It's lower.
```

```
Your guess? 65
It's higher.
Your guess? 66
It's higher.
Your guess? 67
You got it right in 7 guesses
Do you want to play again? nope

Overall results:
    total games   = 3
    total guesses = 15
    guesses/game  = 5.0
    best game     = 4
```